Міністерство освіти і науки України

Тернопільський національний технічний університет

імені Івана ПУЛЮЯ

Кафедра програмної інженерії

ІНСТРУКТИВНО-МЕТОДИЧНІ МАТЕРІАЛИ

для лабораторних занять з дисципліни

ОСНОВИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

для напряму підготовки

121 – “Інженерія програмного забезпечення”

Тернопіль
2016

УДК 681.3(07)
ББК 32.973-01
М54

Укладач:
І.В. Бойко, канд. фіз.-мат. наук, доцент

Рецензент:
М.В. Ткач, докт. фіз.-мат. наук, професор

Методичні вказівки розглянуто й затверджено на засіданні кафедри програмної інженерії
протокол № 6 від 17.05.2016р.

Схвалено та рекомендовано до друку на засіданні методичної комісії факультету комп'ютерно-інформаційних систем та програмної інженерії Тернопільського національного технічного університету імені Івана Пулюя
протокол № 6 від 31.03.2016р.

М54 Інструктивно-методичні матеріали ддля лабораторних занять з дисципліни "Основи програмної інженерії" для напряму підготовки 121 – "Інженерія програмного забезпечення" / Укладач : Бойко І.В.. – Тернопіль : Вид-во ТНТУ імені Івана Пулюя, 2016 – 41 с.

УДК 681.3(07)
ББК 32.973-01

Відповідальний за випуск: *М.Р. Петрик*, докт. фіз.-мат. наук, професор

ЗМІСТ

ВСТУП

Мета дисципліни " Основи програмної інженерії (CS106) ": формування системи теоретичних знань і практичних навичок з зі спеціальності, шляхом застосування знань та навичок з дисципліни “Основи програмування” та найпростіших методів моделювання та застосування даних.

У результаті вивчення дисципліни студент повинен:

Знати:

- основні поняття, методи, алгоритми і моделі апарату інженерії програмного забезпечення;
- властивості моделей, їх синтез та аналіз, види операцій; що можна з ними виконувати;
- фундаментальні поняття, математичні методи і програмні засоби для побудови, реалізації та аналізу найпростіших моделей;
- застосовування технологій програмування та програмного забезпечення для розробки програмних продуктів.

Вміти:

- володіти методами та технологіями організації та застосування програмних засобів для реалізації складних задач;
- оперувати методами за програмними засобами для їх ефективного застосування в найпростіших моделях та задачах ;
- застосовувати набуті знання та вміння для розробки програмних продуктів.

Знання, вміння та навички, отримані студентами під час вивчення даної дисципліни, будуть базовими для подальших вивчення курсу “Основи програмної інженерії ” та курсів, пов’язаних зі моделювання та складанням алгоритмів роботи.

Подаються завдання для лабораторних робіт, приклади їх виконання з допомогою мови програмування С++ та приклади оформлення готових робіт.

Матеріали призначені для полегшення засвоєння дисципліни " Основи програмної інженерії " під час виконання лабораторних занять. Складається з типових завдань, рекомендованих до виконання на лабораторних заняттях з дисципліни “Основи програмної інженерії” під час першого семестру навчання.

Приводяться приклади виконання завдань лабораторних робіт мовою С++ та приклади оформлення отриманих результатів.

Автор щиро вдячний студентам групи СП-11 за надані приклади виконання лабораторних робіт.

ЛАБОРАТОРНА РОБОТА №1

“КАЛЬКУЛЯТОР У РІЗНИХ СИСТЕМАХ ЧИСЛЕННЯ”

Необхідно реалізувати роботу найпростішого калькулятора. Програма вимагає від користувача введення вибору системи числення у якій він бажає працювати, далі деякого числового ряду, який потім обчислює. Якщо обчислення вдале, програма повинна вивести результат. Якщо обчислення не є вдалим, то програма повинна вивести інформацію про помилку. Після виведення інформації програма питає у користувача, чи бажає він продовжити роботу. Якщо так, тоді знову вимагає введення рядку. Якщо ні – програма завершує роботу.

Типи можливих помилок: невідомий символ, не вистачає дужки, ділення на 0.

Можливі варіанти роботи калькулятора у різних системах числення:

1. Калькулятор працює з цілими числами в 10-й системі числення та наступними операціями: +, -, *, / (ціла частина від ділення), % (остача від ділення).
2. Калькулятор працює з цілими числами в 2-й системі числення та наступними операціями: +, - , *, / (ціла частина від ділення), % (остача від ділення).
3. Калькулятор працює з цілими числами в 8-й системі числення та наступними операціями: +, -, *, / (ціла частина від ділення), % (остача від ділення).
4. Калькулятор працює з цілими числами в 16-й системі числення та наступними операціями: +, -, *, / (ціла частина від ділення), % (остача від ділення).
5. Калькулятор працює з дійсними числами в 10-й системі числення та наступними операціями: +, -, *, /.
6. Калькулятор працює з дійсними числами в 10-й системі числення та наступними операціями: +, - , *, /, ^ (піднесення до степеня).
7. Калькулятор працює з цілими числами в 10-й системі числення та наступними операціями: +, - , *, / (ціла частина від ділення), % (остача від ділення), ! (факторіал).
8. Калькулятор працює з цілими числами в 2-й системі числення та наступними операціями: +, - , *, / (ціла частина від ділення), % (остача від ділення), !.
9. Калькулятор працює з дійсними числами в 10-й системі числення та наступними операціями: +, - , *, /, pi (константа π), sin, cos, tg, ctg.

Приклад виконання роботи мовою С++

```
#include <iostream>
#include <Windows.h>
```

```
#include <string.h>
#include <cctype>
#include <math.h>
using namespace std;

char* pattern[9] = { "+-*/%", "+-*/%", "+-*/%", "+-*/%abcdefABCDEF", "+-*/.", "+-*/^.", "+-*/%!",
"+-*/%", "+-*/%.sincotgp" };
int c_syst = 10;

double strtodouble(char*);
int strcnvtoint(char*, int);
int chrcnvtoint(char);
char* intcnvtosyst(long long);

int priority(char);
void clearstack(void);
void check(char*, int);
bool ishex(char);
void fact(long long&);
void error(int);
bool isnumber(char);
bool contain_func(char*, int&, int&);

void operation_l(long long&, long long&, char);
void operation_f(double&, double&, char);
void to_postfix(char*);
char* solve(void);
double solve_f(void);
double solve_wf(char*);


struct {
	char smb[256];	int cur_ind = -1;
	char getop(void) { return smb[cur_ind--]; }
	void pushop(char f) { smb[++cur_ind] = f; }
	char watchop(void) { return smb[cur_ind]; }
} fnstack;

struct {
	char op[256];	int cur_ind = -1;
	char getop(void) { return op[cur_ind--]; }
	void pushop(char c) { op[++cur_ind] = c; }
	char watchop(void) { return op[cur_ind]; }
} opstack;

struct {
	long long val[256]; int cur_ind = -1;
	void operation(char op) {
		if (op != '!')	operation_l(val[--cur_ind], val[cur_ind + 1], op);
		else fact(val[cur_ind]);
	}
	long long getnum(void) { return val[cur_ind--]; }
	void pushnum(long long f) { val[++cur_ind] = f; }
	long long watchop(void) { return val[cur_ind]; }
} valstack;

struct {
	double val[256]; int cur_ind = -1;
	void operation(char op) { operation_f(val[--cur_ind], val[cur_ind + 1], op); }
	double getnum(void) { return val[cur_ind--]; }
	void pushnum(double f) { val[++cur_ind] = f; }
	double watchop(void) { return val[cur_ind]; }
} valfstack;

int main() {
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
```

```
	int r;
	while (true) {
		clearstack();
		system("cls");
		cout << " <- Введіть режим числення (0 - вихід) : \n  -> ";
		cin >> r;
		if (r == 0) exit(0);
		else if (r < 1 || r>9)
			error(3);
		switch (r) {
		case 1:	case 5:	case 6:	case 7: case 9:	c_syst = 10; break;
		case 2:	case 8:	c_syst = 2;	break;
		case 3:	c_syst = 8;	break;
		case 4:	c_syst = 16; break;
		}
		char *a = new char[256];
		cout << " <- Введіть вхідний рядок для обчислень : \n  -> ";
		cin >> a;
		check(a, r);
		if (r != 9) {
			to_postfix(a);
			cout << " <- Рядок, згенерований програмою за алгоритмом ОПЗ :\n  <- " <<
fnstack.smb;
		}
		cout << "\n <- Відповідь : ";
		if (r == 5 || r == 6)
			cout << solve_f() << endl;
		else if (r == 9)
			solve_wf(a);
		else
			cout << solve() << endl;
		system("pause > nul");
	}
	return 0;
}

void to_postfix(char* c) {
	opstack.pushop('|');
	bool op = false;
	for (int i = 0; i < strlen(c) + 1; i++) {
		if (isdigit(c[i]) || c[i] == '.' || ishex(c[i])) {
			if (op)
				fnstack.pushop(' ');
			fnstack.pushop(c[i]);
			op = false;
		}
		else if (c[i] == ')') {
			while (opstack.watchop() != '(') {
				fnstack.pushop(' ');
				fnstack.pushop(opstack.getop());
			}
			op = true;
			opstack.cur_ind--;
		}
		else if (c[i] == '(')
			opstack.pushop('(');
		else if (c[i] == '\0')
			while (opstack.watchop() != '|') {
				fnstack.pushop(' ');
				fnstack.pushop(opstack.getop());
			}
		else {
			op = true;
			if (priority(opstack.watchop()) < priority(c[i]) && opstack.watchop() == '|')
				opstack.pushop(c[i]);
			else {
```

```
                        while (priority(c[i]) <= priority(opstack.watchop()) &&
opstack.watchop() != '|') {
                                fnstack.pushop(' ');
                                fnstack.pushop(opstack.getop());
                        }
                        opstack.pushop(c[i]);
                }
        }
    }
}

bool isnumber(char c) {
    char a = '0';
    for (int i = 0; i < 10; i++) {
        if (c == a) return true; a++;
    } return false;
}
bool ishex(char a) {
    char b = 'a', B = 'A';
    for (int i = 0; i < 6; i++) { if (a == b || a == B) return true; b++; B++; }
    return false;
}

int strcnvtoint(char* a, int s) {
    int p = 0;
    while (isnumber(*a) || ishex(*a)) { p *= s;     p += chrcnvtoint(*a); a++; }
    return p;
}

int chrcnvtoint(char c) {
    if (c >= '0' && c <= '9') return c - '0';
    else
        switch (c) {
        case 'a': case 'A': return 10;
        case 'b': case 'B': return 11;
        case 'c': case 'C': return 12;
        case 'd': case 'D': return 13;
        case 'e': case 'E': return 14;
        case 'f': case 'F': return 15;
        }
}

char intcnvtochr(int n) {
    if (n >= 0 && n <= 9) return n + '0';
    else
        switch (n) {
        case 10: return 'A';
        case 11: return 'B';
        case 12: return 'C';
        case 13: return 'D';
        case 14: return 'E';
        case 15: return 'F';
        }
}

char* intcnvtosyst(long long n) {
    if (n == 0)
        return "0";
    char *a = new char[35];
    int i = 0, k; bool minus = false;
    minus = n < 0;
    n = abs(n);
    while (n != 0) {
        a[i] = intcnvtochr(n%c_syst);
        a[i + 1] = '\0';
        n /= c_syst;
        i++;
```

```
	}
	if (minus) a[i] = '-', i++, a[i] = '\0';
	i = 0, k = strlen(a) - 1;
	for (; i < k; i++, k--) {
		char t = a[i]; a[i] = a[k]; a[k] = t;
	}
	return a;
}

double strtodouble(char* a) {
	double x = 0, b = 1;
	bool point = false;
	while (isdigit(*a) || *a == '.') {
		if (*a == '.') point = true;
		else if (point) {
			b /= 10.;
			x += (*a - '0')*b;
		}
		else {
			x *= 10;
			x += *a - '0';
		}
		a++;
	}
	return x;
}

char* doubletostr(double n) {
	long long a = n; n -= a;
	char g[30] = "";
	strcat(g, intcnvtosyst(a));
	int k = strlen(g)+1;
	g[k-1] = '.';
	for (int i = k; i < k + 6; i++) {
		n *= 10;
		g[i] = intcnvtochr(((int)n) % 10);
	}
	g[k + 6] = '\0';
	cout << " = " <<g;
	return g;
}

int priority(char c) {
	switch (c) {
	case '+':case '-':return 0;
	case '*':case '/':case '%':return 1;
	case '^':case '!':return 2;
	default:return -1;
	}
}

void error(int code) {
	cout << "\n  !- Помилка. ";
	switch (code) {
	case 0:
		cout << "Вираз містить неприпустимі символи або операції"; break;
	case 1:
		cout << "Різна кількість відкритих та закритих дужок"; break;
	case 2:
		cout << "Ділення на нуль"; break;
	case 3:
		cout << "Невідомий режим роботи"; break;
	case 4:
		cout << "Некоректний вираз"; break;
	}
	system("pause > nul");
	exit(0);
```

```
}

void check(char* exp, int n) {
	char* b = new char[sizeof(exp)];
	strcpy(b, exp);
	char c[30] = "\0";
	switch (c_syst) {
	case 10: case 16: strcpy(c,"0123456789()") ; break;
	case 2: strcpy(c, "01()"); break;
	case 8: strcpy(c, "012345678()"); break;
	}
	strcat(c, pattern[n - 1]);
	if (strtok(exp, c) != NULL)
		error(0);
	else {
		int od = 0;
		for (int i = 0; i < strlen(exp); i++) {
			if (exp[i] == '(') od++;
			if (exp[i] == ')') od--;
			if (od < 0) error(1);
		}
		if (od)
			error(1);
	}
}

void fact(long long &n) { int k = 1; for (int i = 2; i <= n; i++) k *= i; n = k; }

char* solve(void) {
	char *c = fnstack.smb;
	while (*c != '\0') {
		if (*c == ' ') c++;
		else if (isnumber(*c) || ishex(*c)) {
			valstack.pushnum(strcnvtoint(c, c_syst));
			while (isnumber(*c) || ishex(*c))
				c++;
		}
		else valstack.operation(*c), c++;
	}
	if (valstack.cur_ind == 0) return intcnvtosyst(valstack.getnum());
	else error(4);
}

double solve_f(void) {
	char *c = fnstack.smb;
	while (*c != '\0') {
		if (*c == ' ') c++;
		else if (isnumber(*c) || (*c) == '.' || ishex(*c)) {
			valfstack.pushnum(strtodouble(c));
			while (isnumber(*c) || (*c) == '.' || ishex(*c))
				c++;
		}
		else valfstack.operation(*c), c++;
	}
	if (valfstack.cur_ind == 0)return valfstack.getnum();
	else error(4);
}

double solve_wf(char* a) {
	char *c = new char[sizeof(a)], *b = new char[sizeof(a)];
	strcpy(c, a);
	char *p = strstr(c, "pi");
	while (p != NULL) {
		strcpy(b, "3.1415");
		strcat(b, c + (p - c) + 2);
		strcpy(c + (p - c), b);
		b[strlen(b) - 1] = '\0';
```

```
		p = strstr(c, "pi");
	}
	cout << c;
	int indl, indr, cd = 0;
	double f;
	while (contain_func(c, indl, indr)) {
		clearstack();
		to_postfix(b);
		f = solve_f();
		strcpy(b, c + indr + 1);
		switch (c[indl - 1]) {
		case 'n':
			strcpy(c + indl - 3, doubletostr(sin(f)));
			break;
		case 's':
			strcpy(c + indl - 3, doubletostr(cos(f)));
			break;
		case 'g':
			if (indl > 2 && c[indl - 3] == 'c')
				strcpy(c + indl - 3, doubletostr(1 / tan(f)));
			else
				strcpy(c + indl - 2, doubletostr(tan(f)));

			break;
		}
	}
	strcat(c, b);
	clearstack();
	to_postfix(c);
	return solve_f();
}

void operation_l(long long& a, long long& b, char c) {
	switch (c) {
	case '+':a = a + b; break;
	case '-':a = a - b; break;
	case '*':a = a * b; break;
	case '/':if (b == 0)error(2);
		 a = a / b; break;
	case '%':a = a % b; break;
	case '^':a = pow(a, b); break;
	}
}

void operation_f(double& a, double& b, char c) {
	switch (c) {
	case '+':a = a + b; break;
	case '-':a = a - b; break;
	case '*':a = a * b; break;
	case '/':if (b == 0) error(2);
		 a = a / b; break;
	case '^':a = pow(a, b); break;
	}
}

void clearstack(void) {
	fnstack.cur_ind = opstack.cur_ind = valfstack.cur_ind = valstack.cur_ind = -1;
}

bool contain_func(char* a, int& l, int& r) {
	int clip = 1;
	for (int i = 0; i < strlen(a); i++) {
		if (a[i] == '(') {
			clip++;
			if (i != 1 && isalpha(a[i - 1]))
				l = i, clip = 1;
		}
```

```
            if (a[i] == ')') clip--;
            if (clip == 0) { r = i; return true; }
        }
        return false;
}
```

```
<- Введіть режим числення (0 - вихід) :
 -> 1
<- Введіть вхідний рядок для обчислень :
 -> 2+5*7-9*(20+5*58)
<- Рядок, згенерований програмою за алгоритмом ОПЗ :
 <- 2 5 7 * + 9 20 5 58 * + * -
<- Відповідь : -2753
```

```
<- Введіть режим числення (0 - вихід) :
 ->  <- Введіть вхідний рядок для обчислень :
 ->  <- Рядок, згенерований програмою за алгоритмом ОПЗ :
 <- FF FF +
<- Відповідь : 1FE
```

```
<- Введіть режим числення (0 - вихід) :
 -> 7
<- Введіть вхідний рядок для обчислень :
 -> (28%6)!
<- Рядок, згенерований програмою за алгоритмом ОПЗ :
 <- 28 6 % !
<- Відповідь : 24
```

```
<- Введіть режим числення (0 - вихід) :
 -> 9
<- Введіть вхідний рядок для обчислень :
 -> sin(pi/6)

<- Відповідь : sin(3.1415/6) = 0.499986
```

ЛАБОРАТОРНА РОБОТА №2

“ АНТРОПОМЕТРІЯ ”

Необхідно створити список значень росту та маси людини у залежності від її статі. Величини росту людини мають бути протабульованими починаючи від 140 до 210 см з кроком 10 см. При цьому для кожного утвореного діапазону росту існує значення ваги людини у залежності від її статі.

1. Скласти програму, яка у вхідних даних:

- на першому кроці запитує стать людини;
- на другому кроці запитує ріст людини;
- виводить значення нормального ваги людини, що відповідає її росту та статі.

2. Скласти програму, що запитує послідовно стать, ріст та вагу людини.

- у випадку відхилення ваги людини при заданому її рості виводить повідомлення, про те що вага більша чи менша від норми, та якою вона має бути.
- вивести величину маси яку повинна набрати людина чи на яку вона має схуднути, щоб її маса було у межах норми.

3. Відома маса та ріст кожної людини, з деякої групи людей. Людей, що мають відхилення від норми у масі при заданому рості будемо умовно називати повними чи худими. Визначити кількість повних людей і середню масу решти людей.

Приклад виконання роботи мовою С++

```
#include<iostream>
#include<conio.h>
#include<locale.h>
#include<string.h>
#include<iomanip>
#include<stdlib.h>
using namespace std;
const int n = 5;
void func_1(float*, float*);
void func_2(float*, float*);
void func_3(float*, float*);
void tabl(float*, float*);
int main() {
	setlocale(0, "");
	int number = 1;
	float masa_b[71];
	float masa_g[71];
	float m_b = 42, m_g = 38;
	for (int i = 0; i <= 70; i++) {
		masa_b[i] = m_b;
		m_b = m_b + 0.73;
	}
	for (int i = 0; i <= 70; i++) {
		masa_g[i] = m_g;
		m_g = m_g + 0.73;
	}
```

```
	while (number != 0) {
		cout << "Меню:\n1-Відповідність маси від росту\n2-Оцінка маси\n3-Характеристика групи\n4-Таблиці відповідності\n0-Вихід\nВаш вибір: ";
		cin >> number;
		switch (number) {
		case 1:func_1(masa_b, masa_g); break;
		case 2:func_2(masa_b, masa_g); break;
		case 3:func_3(masa_b, masa_g); break;
		case 4:tabl(masa_b, masa_g); break;
		case 0:break;
		default:cout << "Помилка вибору!" << endl; break;
		}
		cout << "Продовжити роботу?(1-так 0-ні): ";
		cin >> number;
		system("cls");
	}
	return 0;
}
void func_1(float* masa_b, float* masa_g) {
	char stat;
	int srist;
	cout << "Введіть стать(b-(чол.),g-(жін.)): ";
	cin >> stat;
	if (stat == 'b') {
		cout << "Введіть ріст: ";
		cin >> srist;
		cout << "Нормальна маса: " << setprecision(3) << masa_b[srist - 140] << "кг" << endl;
	}
	else {
		if (stat == 'g') {
			cout << "Введіть ріст: ";
			cin >> srist;
			cout << "Нормальна маса: " << setprecision(3) << masa_g[srist - 140] << "кг" << endl;
		}
		else
			cout << "Помилка" << endl;
	}
}
void func_2(float* masa_b, float* masa_g) {
	char stat;
	int srist;
	float masa;
	cout << "Введіть стать(b-(чол.),g-(жін.)): ";
	cin >> stat;
	if (stat == 'b') {
		cout << "Введіть ріст: ";
		cin >> srist;
		cout << "Введіть вагу: ";
		cin >> masa;
		if (masa>masa_b[srist - 140]) {
			cout << "Маса людини більша за нормальну!" << endl << "Нормальна маса: " << setprecision(3) << masa_b[srist - 140] << "кг" << endl;
			cout << "Потрібно схуднути на " << setprecision(3) << masa - masa_b[srist - 140] << "кг" << endl;
		}
		else {
			if (masa<masa_b[srist - 140]) {
				cout << "Маса людини менша за нормальну!" << endl << "Нормальна маса: " << setprecision(3) << masa_b[srist - 140] << "кг" << endl;
				cout << "Потрібно набрати: " << setprecision(3) << masa_b[srist - 140] - masa << "кг" << endl;
			}
			else cout << "Маса людини в нормі!" << endl;
		}
	}
```

```
	else {
		if (stat == 'g') {
			cout << "Введіть ріст: ";
			cin >> srist;
			cout << "Введіть вагу: ";
			cin >> masa;
			if (masa>masa_g[srist - 140]) {
				cout << "Маса людини більша за нормальну!" << endl << "Нормальна маса:
" << setprecision(3) << masa_g[srist - 140] << "кг" << endl;
				cout << "Потрібно схуднути на " << setprecision(3) << masa -
masa_g[srist - 140] << "кг" << endl;
			}
			else {
				if (masa<masa_g[srist - 140]) {
					cout << "Маса людини менша за нормальну!" << endl << "Нормальна
маса: " << setprecision(3) << masa_g[srist - 140] << "кг" << endl;
					cout << "Потрібно набрати: " << setprecision(3) << masa_g[srist
- 140] - masa << "кг" << endl;
				}
				else cout << "Маса людини в нормі!" << endl;
			}
		}
		else
			cout << "Помилка" << endl;
	}
}
void func_3(float* masa_b, float* masa_g) {
	char stat;
	int srist, big_p = 0;
	float masa, s_masa = 0;
	cout << "Група складається з 5 чоловік!" << endl;
	for (int i = 0; i<n; i++) {
		cout << "Введіть стать " << i + 1 << " людини(b-(чол.),g-(жін.)): ";
		cin >> stat;
		if (stat == 'b') {
			cout << "Введіть ріст: ";
			cin >> srist;
			cout << "Введіть вагу: ";
			cin >> masa;
			if (masa>masa_b[srist - 140])
				big_p++;
			else
				s_masa += masa;
		}
		else {
			if (stat == 'g') {
				cout << "Введіть ріст: ";
				cin >> srist;
				cout << "Введіть вагу: ";
				cin >> masa;
				if (masa>masa_g[srist - 140])
					big_p++;
				else
					s_masa += masa;
			}
			else cout << "Помилка" << endl;

		}

	}
	cout << "Кількість повних людей: " << big_p << endl;
	if (n == big_p)
		cout << "Всі люди повні! Середня маса інших людей відсутня!";
	else
		cout << "Середня маса не повних людей: " << setprecision(3) << s_masa / (n - big_p)
<< "кг" << endl;
}
```

```
void tabl(float* masa_b, float* masa_g) {


	cout << "Відповідність маси і росту(чол.)" << endl << " _________________" << endl << "|";
	for (int i = 0; i <= 70; i += 10) {
		cout << "(" << i + 140 << "см) =" << setw(4) << setprecision(3) << masa_b[i] << "кг|
";
		if (i % 10 == 0)cout << endl << "|";
	}
	cout << "_________________|" << endl;
	cout << endl;
	cout << "Відповідність маси і росту(жін.)" << endl << " _________________" << endl << "|";
	for (int i = 0; i <= 70; i += 10) {
		cout << "(" << i + 140 << "см) =" << setw(4) << setprecision(3) << masa_g[i] << "кг|
";
		if (i % 10 == 0)cout << endl << "|";
	}
	cout << "_________________|" << endl;
}
```

```
1-Відповідність маси від росту
2-Оцінка маси
3-Характеристика групи
4-Таблиці відповідності
0-Вихід
Ваш вибір: 1
Введіть стать(b-(чол.),g-(жін.)): b
Введіть ріст: 182
Нормальна маса: 72.7кг
Продовжити роботу?(1-так 0-ні):
```

ЛАБОРАТОРНА РОБОТА №3

“ ГЕОМЕТРИЧНІ ФІГУРИ”

Частина 1. Планіметрія.

Скласти програму, яка за уведеними параметрами геометричних фігур обчислює їх периметри, площі, кути між сторонами. У програмі повинні реалізовуватися можливості введення сторін планіметричних фігур та кутів між ними.

Програма передбачує на вході запитує назву геометричної фігури і виконує такі дії:

- запитує спосіб обчислення площі трикутника, чи інші дії;
- згідно вибраного методу обчислення запитує вхідні дані;
- за введеними сторонами трикутника обчислює периметр та площу трикутника;
- обчислює площу трикутника за сторонами і кутом між ними;
- реалізує обчислення кутів за теоремою косинусів;
- перевіряє, чи є трикутник рівностороннім, рівнобедреним чи прямокутним;
- виконує обчислення площ паралелограма, ромба, прямокутника, квадрата.

Частина 2. Стереометрія.

Скласти програму, яка на вході запитує тип стереометричної фігури і виконує такі обчислення:

- запитує геометричні параметри стереометричної фігури:
- знаходить об’єм призми;
- знаходить об’єм похилої призми;
- знаходить об’єм циліндра;
- знаходить об’єм конуса (в тому числі зрізаного);
- знаходить об’єм тетраедра;
- у випадку знаходження об’єму тетраедра реалізувати дві можливості:
- знаходження об’єму тетраедра за його сторонами;
- знаходження об’єму тетраедра за координатами його вершин.

Приклад виконання роботи мовою С++

```
#include "stdafx.h"
#include "iostream"
#include "Windows.h"
#include "math.h"

using namespace std;
#define PI 3.14159265
```

```
struct point {
	float x;
	float y;
	float z;
};

void f1(int);
void f2(int);
float* input(int);
float* input1(int);
point* input_p();
bool per(float*a);


int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	int zad;
	while (true) {
		system("cls");
		cout << "Вв вид задачі:\n1|Планіметрія\n2|Стереометрія\n0|вихід з програми\n";
		cin >> zad;
		switch (zad) {
		case 1: {
			while (true) {
				system("cls");
				int fig;
				cout << "Вв назву
фігури\n1|трикутник\n2|паралелограм\n3|ромб\n4|прямокутник\n5|квадрат\n-1|попереднє меню\n0|вихід
з програми\n";
				cin >> fig;
				if (fig == -1) break;
				switch (fig) {
				case 1: {
					while (true) {
						system("cls");
						int r;
						cout << "\tmenu:\n1|обчислення периметра і площі за
героном\n2|обчислення площі за сторонами і кутом між ними\n";
						cout << "3|обчислення кутів за теоремою косинусів\n4|вид
трикутника\n-1|попереднє меню\n0|вихід з програми\n";
						cin >> r;
						if (r == -1) break;
						switch (r) {
						case 1: f1(11); break;
						case 2: f1(12); break;
						case 3: f1(13); break;
						case 4: f1(14); break;
						case 0: exit(1);
						default: {cout << "Некоректне введення\n";
system("pause"); };
						}

					}
					break;
				}
				case 2: f1(2); break;
				case 3: f1(3); break;
				case 4: f1(4); break;
				case 5: f1(5); break;
				case 0: exit(1); break;
				default: {cout << "Некоректне введення\n"; system("pause"); };
				}

			} break;
		}
```

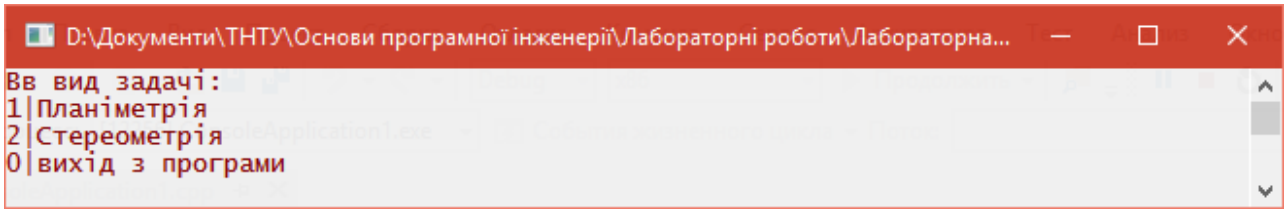
```
		case 2: {
			while (true) {
				system("cls");
				int fig;
				cout << "Вв назву фігури\n1|призма\n2|похила
призма\n3|циліндр\n4|конус\n5|зрізаний конус\n6|тетраедр(за сторонами)\n7|тетраедр(за точками)\n-
1|попереднє меню\n0|вихід з програми\n";
				cin >> fig;
				if (fig == -1) break;
				switch (fig) {
				case 1: f2(1); break;
				case 2: f2(2); break;
				case 3: f2(3); break;
				case 4: f2(4); break;
				case 5: f2(5); break;
				case 6: f2(6); break;
				case 7: f2(7); break;
				case 0: exit(1);
				default: {cout << "Некоректне введення\n"; system("pause"); };
				}

			} break;
		}

		case 0: exit(1);
		default: {cout << "Некоректне введення\n"; system("pause"); };
		}
	}


	system("pause");
	return 0;
}

void f1(int n) {
	switch (n) {
	case 11: {
		float *a = input(1);
		if (!per(a)) { cout << "Невірне введення!" << endl; system("pause"); return; };
		float P = a[0] + a[1] + a[2];
		float p = (float)P / 2;
		double S = sqrt(p*(p - a[0])*(p - a[1])*(p - a[2]));
		cout << "S = " << S << "\tP = " << P << endl;
		system("pause");
		return;
	}
	case 12: {
		float *a = input(2);
		double S = 0.5 * a[0] * a[1] * sin(a[2] * PI / 180);
		cout << "S = " << S << endl;
		system("pause");
		return;
	}
	case 13: {
		float *a = input(1);
		if (!per(a)) { cout << "Невірне введення!" << endl; system("pause"); return; };
		double A = acos(-(a[0] * a[0] - a[1] * a[1] - a[2] * a[2]) / (double)(2 * a[1] *
a[2])) * 180 / PI;
		double B = acos(-(a[1] * a[1] - a[0] * a[0] - a[2] * a[2]) / (double)(2 * a[0] *
a[2])) * 180 / PI;
		double C = 180 - A - B;
		cout << "A = " << A << "\tB = " << B << "\tC = " << C << endl;
		system("pause");
		return;
	}
	case 14: {
		float *a = input(1);
```

```
			if (!per(a)) { cout << "Невірне введення!" << endl; system("pause"); return; };
			int max = 0;
			if (a[max] < a[1]) max = 1;
			if (a[max] < a[2]) max = 2;
			float sk = 0;
			for (int i = 0; i < 3; i++)
				if (i != max) sk += a[i] * a[i];
			if (pow(a[max], 2) == sk) { cout << "Прямокутний\n"; system("pause"); return; }
			if (a[0] == a[1] && a[0] == a[2]) { cout << "Рівносторонній\n"; system("pause");
return; }
			if (a[0] == a[1] || a[0] == a[2] || a[1] == a[2]) { cout << "Рівнобедренний\n";
system("pause"); return; }
			return;
		}
		case 2: {
			float *a = input(3);
			double S = a[0] * a[1];
			cout << "S = " << S << endl;
			system("pause");
			return;
		}
		case 3: {
			float *a = input(4);
			double S = 0.5 * a[0] * a[1];
			cout << "S = " << S << endl;
			system("pause");
			return;
		}
		case 4: {
			float *a = input(5);
			double S = a[0] * a[1];
			cout << "S = " << S << endl;
			system("pause");
			return;
		}
		case 5: {
			system("cls");
			cout << "Введіть сторону квадрата ";
			float a;
			cin >> a;
			double S = a*a;
			cout << "S = " << S << endl;
			system("pause");
			return;
		}

		}

}

void f2(int n) {
		switch (n) {
		case 1: {
			float *a = input1(1);
			double V = a[0] * a[1];
			cout << "V = " << V << endl;
			system("pause");
			return;
		}
		case 2: {
			float *a = input1(2);
			double V = a[0] * a[1];
			cout << "V = " << V << endl;
			system("pause");
			return;
		}
		case 3: {
```

```
			float *a = input1(3);
			double V = (a[0] * a[0] * PI)*a[1];
			cout << "V = " << V << endl;
			system("pause");
			return;
		}
		case 4: {
			float *a = input1(4);
			double V = (a[0] * a[0] * a[1] * PI) / 3;
			cout << "V = " << V << endl;
			system("pause");
			return;
		}
		case 5: {
			float *a = input1(5);
			double V = ((a[0] * a[0] + a[0] + a[1] + a[1] * a[1]) * a[2] * PI) / 3;
			cout << "V = " << V << endl;
			system("pause");
			return;
		}
		case 6: {
			float *a = input1(6);
			if (!per(a)) { cout << "Невірне введення!" << endl; system("pause"); return; };
			float p = a[0] + a[1] + a[2];
			p /= 2;
			double V = (sqrt(p*(p - a[0])*(p - a[1])*(p - a[2]))*a[3]) / 3;
			cout << "V = " << V << endl;
			system("pause");
			return;
		}
		case 7: {
			point *a = input_p();
			double V = (a[0].x*a[1].y*a[2].z) + (a[1].x*a[2].y*a[0].z) + (a[0].y*a[1].z*a[2].x)
- (a[0].z*a[1].y*a[2].x) - (a[1].x*a[0].y*a[2].z) - (a[2].y*a[1].z*a[0].x);
			V /= 6;
			cout << "V = " << V << endl;
			system("pause");
			return;
		}


		}
}

float* input(int r) {
		switch (r) {
		case 1: {
			float *a = new float[3];
			system("cls");
			cout << "Введіть три сторон трикутника ";
			cin >> a[0] >> a[1] >> a[2];
			return a;
		}
		case 2: {
			float *a = new float[3];
			system("cls");
			cout << "Введіть дві сторони трикутника і кут між ними ";
			cin >> a[0] >> a[1] >> a[2];
			return a;
		}
		case 3: {
			float *a = new float[2];
			system("cls");
			cout << "Введіть сторону і висоту паралелограма ";
			cin >> a[0] >> a[1];
			return a;
		}
```

```
	case 4: {
		float *a = new float[2];
		system("cls");
		cout << "Введіть діагоналі ромба ";
		cin >> a[0] >> a[1];
		return a;
	}
	case 5: {
		float *a = new float[2];
		system("cls");
		cout << "Введіть сторони прямокутника ";
		cin >> a[0] >> a[1];
		return a;
	}


	}

	return 0;
}

float* input1(int r) {
	switch (r) {
	case 1: {
		float *a = new float[2];
		system("cls");
		cout << "Введіть площу основи і висоту призми ";
		cin >> a[0] >> a[1];
		return a;
	}
	case 2: {
		float *a = new float[2];
		system("cls");
		cout << "Введіть площу основи і висоту похилої призми ";
		cin >> a[0] >> a[1];
		return a;
	}
	case 3: {
		float *a = new float[2];
		system("cls");
		cout << "Введіть радіус основи і висоту циліндра ";
		cin >> a[0] >> a[1];
		return a;
	}
	case 4: {
		float *a = new float[2];
		system("cls");
		cout << "Введіть радіус основи і висоту конуса ";
		cin >> a[0] >> a[1];
		return a;
	}
	case 5: {
		float *a = new float[3];
		system("cls");
		cout << "Введіть великий і малий радіуси основ і висоту зрізаного конуса ";
		cin >> a[0] >> a[1] >> a[2];
		return a;
	}
	case 6: {
		float *a = new float[4];
		system("cls");
		cout << "Введіть три сторони основи і висоту тетраедра ";
		cin >> a[0] >> a[1] >> a[2] >> a[3];
		return a;
	}

	}
```

```
	return 0;
}

point* input_p() {
	point *a = new point[4];
	system("cls");
	cout << "Введіть координати точок тетраедра ";
	for (int i = 0; i < 4; i++) {
		cout << "x" << i + 1 << " = ";
		cin >> a[i].x;
		cout << "y" << i + 1 << " = ";
		cin >> a[i].y;
		cout << "z" << i + 1 << " = ";
		cin >> a[i].z;
	}
	point vec[3];
	vec[0].x = a[1].x - a[0].x;
	vec[0].y = a[1].y - a[0].y;
	vec[0].z = a[1].z - a[0].z;
	vec[1].x = a[2].x - a[0].x;
	vec[1].y = a[2].y - a[0].y;
	vec[1].z = a[2].z - a[0].z;

	vec[1].x = a[3].x - a[0].x;
	vec[1].y = a[3].y - a[0].y;
	vec[1].z = a[3].z - a[0].z;

	return vec;
}

bool per(float *a) {
	if ((a[0] + a[1] > a[2]) && (a[1] + a[2] > a[0]) && (a[0] + a[2] > a[1])) return true;
	return false;
}
```



ЛАБОРАТОРНА РОБОТА №4

“ ОПЕРАТОР ПОЛЬОТУ КОСМІЧНОГО КОРАБЛЯ”

1. Для початку процедури запуску космічного корабля його оператор має увести команду – пароль, що складається з латинських символів та цифр. Зчитування пароля відбувається посимвольно, тобто при введенні пароля перший його символ порівнюється з його відповідником у системі. При співпадінні символів можна вводити наступний символ. Якщо символи не співпадають, то пароль необхідно вводити спочатку.

Завдання: скласти програму яка реалізує згаданий алгоритм введення пароля.

2. Під час виведення космічного корабля на орбіту його швидкість змінюється за законом:

$$\upsilon = \upsilon_0 \ln\left(\frac{M}{M-m}\right), \qquad (1)$$

де M - початкова маса космічного корабля з паливом, m - маса палива, υ_0 - початкова швидкість корабля. При цьому висота підняття корабля залежить від часу як $h = \upsilon t \cos\theta$, де $0 \le \theta \le \frac{\pi}{2}$ - кут атаки корабля при виході на орбіту.

Маса палива змінюється за законом: $m = m_0\left(1 - \frac{t}{\tau}\right)$, де m_0 - початкова маса палива, τ - час польоту.

Корабель якийсь час розганяється і при виході на заплановану висоту h - гальмує.

Нехай нам потрібно вивести на орбіту корабель масою M, що має запас палива m на орбіту, що має висоту h, досягнувши швидкості υ. Корабель має n_1 двигунів, що працюють до моменту t_1 досягнення кораблем певної швидкості $\upsilon_1 = \upsilon/10$, далі працює наступний каскад двигунів n_2 протягом часу $t_2/5$ і так далі до досягнення необхідної висоти. При цьому повинна подаватися команда до гальмування.

Завдання 1: скласти програму, що за введеними значеннями M, m, h, υ керує роботою двигунів табулюючи значення швидкості та висоти корабля, видаючи повідомлення про досягнення відповідних висоти та швидкості, включення чергового каскаду двигунів, початок процесу гальмування.

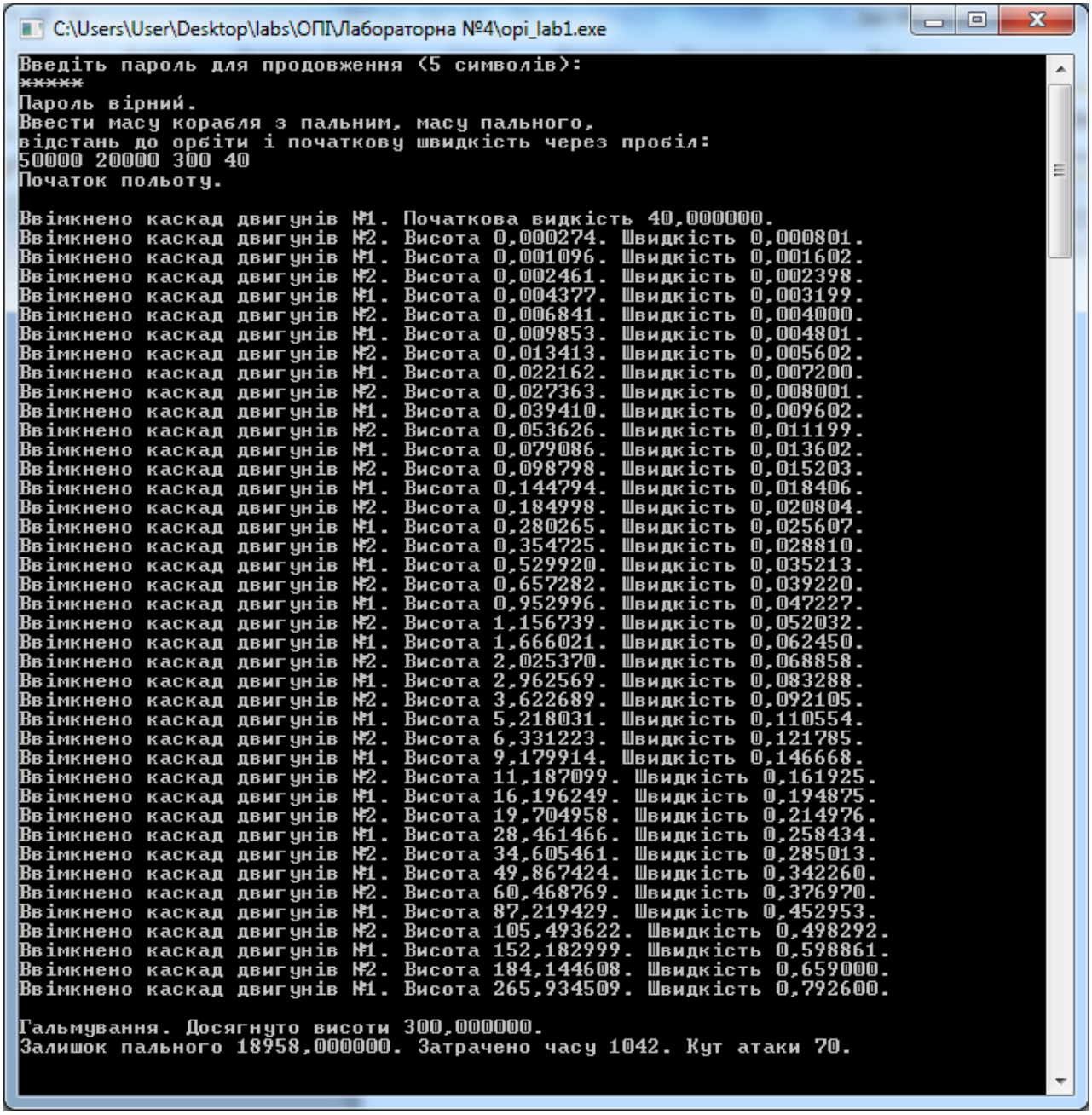
Завдання 2: вивести залежності швидкості корабля від його маси, та його висоти від часу з вказаними точками перемикання каскадів двигунів та моменту його гальмування.

Приклад виконання роботи мовою С++

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <locale.h>
#include <windows.h>
#include <fstream>
using namespace std;
const char password[] = "12345";

bool function_login(char p, int i){
if(password[i] == p) return true; else return false;
}

void procedure_login(){
const int n = sizeof(password)-1;
int i = 0;
bool flag = true;

printf("Введіть пароль для продовження (%d символів):\n",n);
for(int i = 0; i < n; i++){
if(!function_login(getch(),i))flag = false;
printf("*");
if(i == n-1 && !flag){
i = -1; flag = true; printf("\nПароль введено невірно. Спробуйте ще раз:\n");
}
}
printf("\nПароль вірний.\n");
}

void procedure_fly(){
ofstream fout("output.csv");
float M, m, h0, v0;
printf("Ввести масу корабля з пальним, масу пального,\nвідстань до орбіти і початкову
швидкість через пробіл:\n");
scanf("%f%f%f%f",&M,&m,&h0,&v0);
srand(time(NULL));
int t, t1, alpha = rand()%90;

float v1 = v0*log(1), h, v = v0;

bool flag = false, flag_h = false;

printf("Початок польоту.\n\nВвімкнено каскад двигунів №1. Початкова видкість %f.\n",v);
for(t = 1; h < h0; t++){
if(t>m){flag_h = false; break;}
v = v0*log(M/(M-t));
h = v*t*cos(alpha*M_PI/180);
if(h>h0) break;
if(!flag && v>=v1*1.1){flag = true; t1 = t;
printf("Ввімкнено каскад двигунів №2. Висота %f. Швидкість %f.\n",h,v);
fout<<v<<";"<<(M-t)<<";"<<h<<";"<<t<<endl;
}else
if(flag && t>=t1*1.2){flag = false; v1 = v;
```

```
printf("Ввімкнено каскад двигунів №1. Висота %f. Швидкість %f.\n",h,v);
fout<<v<<";"<<(M-t)<<";"<<h<<";"<<t<<endl;
}
flag_h = true;
}
if(flag_h){
printf("\nГальмування. Досягнуто висоти %f.\nЗалишок пального %f. Затрачено часу %d. Кут атаки %d.\n",h0,m-t,t,alpha);
fout<<v<<";"<<(M-t)<<";"<<h0<<";"<<t<<endl;
}else{
printf("\nПальне закінчилось. Час падіння %d. Висота %f. Кут атаки %d.\n",t,h,alpha);
fout<<0<<";"<<(M-t)<<";"<<h<<";"<<t<<endl;
}
fout.close();
}

int main(){
setlocale(0, "");
procedure_login();
procedure_fly();
getch();
return 0;
}
```

Результат.


C:\Users\User\Desktop\labs\ОПІ\Лабораторна №4\opi_lab1.exe
Введіть пароль для продовження (5 символів):
*****
Пароль вірний.
Ввести масу корабля з пальним, масу пального,
відстань до орбіти і початкову швидкість через пробіл:
50000 20000 300 40
Початок польоту.
Ввімкнено каскад двигунів №1. Початкова видкість 40,000000.
Ввімкнено каскад двигунів №2. Висота 0,000274. Швидкість 0,000801.
Ввімкнено каскад двигунів №1. Висота 0,001096. Швидкість 0,001602.
Ввімкнено каскад двигунів №2. Висота 0,002461. Швидкість 0,002398.
Ввімкнено каскад двигунів №1. Висота 0,004377. Швидкість 0,003199.
Ввімкнено каскад двигунів №2. Висота 0,006841. Швидкість 0,004000.
Ввімкнено каскад двигунів №1. Висота 0,009853. Швидкість 0,004801.
Ввімкнено каскад двигунів №2. Висота 0,013413. Швидкість 0,005602.
Ввімкнено каскад двигунів №1. Висота 0,022162. Швидкість 0,007200.
Ввімкнено каскад двигунів №2. Висота 0,027363. Швидкість 0,008001.
Ввімкнено каскад двигунів №1. Висота 0,039410. Швидкість 0,009602.
Ввімкнено каскад двигунів №2. Висота 0,053626. Швидкість 0,011199.
Ввімкнено каскад двигунів №1. Висота 0,079086. Швидкість 0,013602.
Ввімкнено каскад двигунів №2. Висота 0,098798. Швидкість 0,015203.
Ввімкнено каскад двигунів №1. Висота 0,144794. Швидкість 0,018406.
Ввімкнено каскад двигунів №2. Висота 0,184998. Швидкість 0,020804.
Ввімкнено каскад двигунів №1. Висота 0,280265. Швидкість 0,025607.
Ввімкнено каскад двигунів №2. Висота 0,354725. Швидкість 0,028810.
Ввімкнено каскад двигунів №1. Висота 0,529920. Швидкість 0,035213.
Ввімкнено каскад двигунів №2. Висота 0,657282. Швидкість 0,039220.
Ввімкнено каскад двигунів №1. Висота 0,952996. Швидкість 0,047227.
Ввімкнено каскад двигунів №2. Висота 1,156739. Швидкість 0,052032.
Ввімкнено каскад двигунів №1. Висота 1,666021. Швидкість 0,062450.
Ввімкнено каскад двигунів №2. Висота 2,025370. Швидкість 0,068858.
Ввімкнено каскад двигунів №1. Висота 2,962569. Швидкість 0,083288.
Ввімкнено каскад двигунів №2. Висота 3,622689. Швидкість 0,092105.
Ввімкнено каскад двигунів №1. Висота 5,218031. Швидкість 0,110554.
Ввімкнено каскад двигунів №2. Висота 6,331223. Швидкість 0,121785.
Ввімкнено каскад двигунів №1. Висота 9,179914. Швидкість 0,146668.
Ввімкнено каскад двигунів №2. Висота 11,187099. Швидкість 0,161925.
Ввімкнено каскад двигунів №1. Висота 16,196249. Швидкість 0,194875.
Ввімкнено каскад двигунів №2. Висота 19,704958. Швидкість 0,214976.
Ввімкнено каскад двигунів №1. Висота 28,461466. Швидкість 0,258434.
Ввімкнено каскад двигунів №2. Висота 34,605461. Швидкість 0,285013.
Ввімкнено каскад двигунів №1. Висота 49,867424. Швидкість 0,342260.
Ввімкнено каскад двигунів №2. Висота 60,468769. Швидкість 0,376970.
Ввімкнено каскад двигунів №1. Висота 87,219429. Швидкість 0,452953.
Ввімкнено каскад двигунів №2. Висота 105,493622. Швидкість 0,498292.
Ввімкнено каскад двигунів №1. Висота 152,182999. Швидкість 0,598861.
Ввімкнено каскад двигунів №2. Висота 184,144608. Швидкість 0,659000.
Ввімкнено каскад двигунів №1. Висота 265,934509. Швидкість 0,792600.
Гальмування. Досягнуто висоти 300,000000.
Залишок пального 18958,000000. Затрачено часу 1042. Кут атаки 70.

Лабораторна робота №5

"Елементарний шаховий рушій"

Нумерація клітинок на шаховій дошці дається за допомогою восьми горизонтальних латинських бука: a, b, c, d, e , f, g, h та восьми послідовних цифр: 1, 3, 4, 5, 6, 7, 8. Існують такі шахові фігури: король, ферзь, тура, слон, кінь та пішак.

Їхні позначення такі: Кр, Ф, Т, С, К, п.

Напочатку шахова дошка вважається чистою – на ній немає жодної фігури.

Завдання 1. Скласти програму, що включає команди розстановки фігур на дошку, наприклад Фe1- означає, що ферзь розміщений на полі e1.

Програма має працювати таким чином:

- на вході запитується назва фігури;
- після введення координати фігури – запитується її координата, у випадку неправильного її введення видається повідомлення про це;
- виводиться повідомлення: наприклад – король розміщений на полі f6;
- якщо при введенні координат фігур було введено однакові координати, то виводиться повідомлення: "це поле вже зайняте";
- командою endgame закінчуємо розстановку фігур і виводимо у стовбець інформацію про положення розміщених фігур.

Завдання 2. Скласти програму, що для розставлених фігур описує можливі ходи.

1. Хід пішака:

- хід пішака можливий лише по вертикалі, і задається командою, що задає координати початкового та кінцевого положень, наприклад: пe2-e4. Початковий хід можливий на дві клітинки, наступні лише на одну.
- якщо шлях пішака зайнятий іншою фігурою – вивести повідомлення про неможливість ходу;
- поле перетворення: при досягненні пішаком полів з координатами: a8, b8, c8, d8, e8 , f8, g8, h8 чи a1, b1, c1, d1, e1 , f1, g1, h1 описати перетворення пішака на одну з фігур: Ф, Т, С, К. Вивести повідомлення про це.

2. Хід коня.

- описати можливі скачки коня у межах квадрата 4*4, задаючи його початкову та кінцеву координати, наприклад: Kf1-f3;

- описати випадки неможливості ходів, коли кінь розміщений на краю дошки;

- описати випадки неможливості ходів, коли поле зайняте.

3. Хід слона.

- у залежності від кольору слона описати його ходи по діагоналі, задаючи початкове та кінцеве положення, наприклад: Cc1-c4.

- описати випадки неможливості ходів, коли поле зайняте.

4. Хід тури.

- описати можливі ходи тури по горизонталі чи вертикалі, задаючи початкову та кінцеві координати, наприклад Th1-h4;

- описати випадки неможливості ходів, коли поле зайняте.

5. Хід ферзя.

- описати можливі ходи ферзя по горизонталі, вертикалі чи діагоналях, задаючи початкову та кінцеві координати;

- описати випадки неможливості ходів, коли поле зайняте.

Завдання 3. Пріоритет та вага фігур.

- Описати вартість кожної з фігур у пішаках наступним чином: кінь, слон – три пішаки, тура – п’ять пішаків, ферзь – 10 пішаків.

- розставити на шахову дошку довільну кількість білих та чорних фігур.

- розрахувати сумарну вартість фігур кожної масті у пішаках;

- вивести повідомлення: у білих (чорних) перевага: +x (пішаків), де x – різниця між сумарними вагами фігур різної масті.

Приклад виконання роботи мовою С++

```
#include <iostream>
#include <Windows.h>
#include <time.h>
#include <iomanip>
using namespace std;

enum ConsoleColor {
	Black = 0, Blue = 1, Green = 2, Cyan = 3, Red = 4, Magenta = 5, Brown = 6, LightGray = 
7, DarkGray = 8,
	LightBlue = 9, LightGreen = 10, LightCyan = 11, LightRed = 12, LightMagenta = 13, 
Yellow = 14, White = 15
};

void SetColor(ConsoleColor text, ConsoleColor background = White) {
	HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);
	SetConsoleTextAttribute(hStdOut, (WORD)((background << 4) | text));
```

```
}

int desk[8][8];
bool coord_(char*, int&);
bool name_(char*, int&);
void input();
void put(int, int, int);
void print();
int get(int);
char* getcoord(int, int);
void moving();
void pishak(int, int);
void kin(int, int);
void slon(int, int);
void tura(int, int);
void ferz(int, int);
void kin_t(int, int, int, int, bool&);
bool coord_t(char*, int&);
void random_fill(bool);
void second_task();

int main() {
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	int mode;
	while (true) {
		system("cls");
		cout << "Введіть режим роботи з програмою:\n  -1 : Розстановка фігур та
прорахування ходів\n" <<
			"  -2 : Випадкова розстановка та обчислення вартості\n  -0 : Вихід\n";
		cin >> mode;
		switch (mode) {
		case 1:
			cin.ignore();
			input(); print();
			system("pause > nul");
			moving();
			system("pause > nul");
			break;
		case 2:
			second_task();
			break;
		case 0:
			exit(0);
		default:
			cout << "Невідомий режим роботи. Спробуйте знову.\n";
			system("pause > nul");
			continue;
		}
	}
	exit(0);
}

void clean() {
	for (int i = 0; i < 8; i++)
		for (int j = 0; j < 8; j++)
			desk[i][j] = 0;
}

void input() {
	bool finish = false;
	clean();
	char *name = new char[41], *coord = new char[41];
	do {
```

```
		cout << "Введіть назву фігури (endgame для завершення): ";
		int n, c;
		gets_s(name, 40);
		if (strcmp(name, "endgame") != 0) {
			while (true) {

				if (strcmp(name, "endgame") == 0) {
					finish = true;
					break;
				}
				else if (!name_(name, n))
					cout << "Помилкова назва. Введіть повторно: ";

				else
					break;
				gets_s(name, 40);
			};
			if (finish) break;
			cout << "Тепер введіть поле, на яке потрібно помістити фігуру: ";
			while (true) {
				gets_s(coord, 40);
				if (!coord_(coord, c))
					cout << "Помилкові координати. Введіть знову : ";
				else
					break;
			};

			put(n, c, 0);
		}
		else
			break;
	} while (true);
}

char* getname(int k, int j) {
	switch (desk[k][j] % 10) {
	case 1:	return "Пішак";
	case 2: return "Кінь";
	case 3: return "Слон";
	case 4: return "Тура";
	case 5: return "Ферзь";
	case 6: return "Король";
	}
	return "";
}

char* getcoord(int k, int j) {
	char* c = new char[3];
	c[0] = ('a' + k);
	c[1] = (j + 1 + '0');
	c[2] = '\0';
	return c;
}

char* figure(int f) {
	switch (f) {
	case 1: return "п ";
	case 2: return "К ";
	case 3: return "С ";
	case 4: return "Т ";
	case 5: return "Ф ";
	case 6: return "Кр";
	case 0: return "  ";
	}
```

```
}

bool name_(char* name, int& n) {
	switch (strlen(name)) {
	case 1:
		switch (name[0]) {
		case 'К':	n = 2; return true;
		case 'Ф':	n = 5; return true;
		case 'Т':	n = 4; return true;
		case 'С':	n = 3; return true;
		case 'п':	n = 1; return true;
		default: return false;
		}
	case 2:
		if (strcmp(name, "Кр") == 0) {
			n = 6;
			return true;
		}
		else
			return false;
	default:
		return false;
	}
}

bool coord_(char* coord, int& c) {
	int x = 0, y = 0;
	switch (coord[0]) {
	case 'A':	case 'a':	x = 1; break;
	case 'B':	case 'b':	x = 2; break;
	case 'C':	case 'c':	x = 3; break;
	case 'D':	case 'd':	x = 4; break;
	case 'E':	case 'e':	x = 5; break;
	case 'F':	case 'f':	x = 6; break;
	case 'G':	case 'g':	x = 7; break;
	case 'H':	case 'h':	x = 8; break;
	default:return false;
	}
	y = coord[1] - '0';
	if (y > 0 && y < 9) {
		if (desk[y - 1][x - 1] == 0)
		{
			c = 10 * y + x;
			return true;
		}
		else {
			cout << ". Це поле вже зайняте";
			return false;
		}
	}
	return false;
}

bool coord_t(char* coord, int& c) {
	int x = 0, y = 0;
	switch (coord[0]) {
	case 'A':	case 'a':	x = 1; break;
	case 'B':	case 'b':	x = 2; break;
	case 'C':	case 'c':	x = 3; break;
	case 'D':	case 'd':	x = 4; break;
	case 'E':	case 'e':	x = 5; break;
	case 'F':	case 'f':	x = 6; break;
	case 'G':	case 'g':	x = 7; break;
	case 'H':	case 'h':	x = 8; break;
```

```
	default:return false;
	}
	y = coord[1] - '0';
	if (y > 0 && y < 9) {
		c = 10 * y + x;
		return true;
	}
	return false;
}

void put(int n, int c, int color) {
	desk[c / 10 - 1][c % 10 - 1] = color * 10 + n;
}

void print() {
	for (int i = 0; i < 8; i++)
		for (int j = 0; j < 8; j++)
			if (desk[i][j] != 0)
				cout << getname(i, j) << " розміщений(на) на полі " << getcoord(j,
i) << endl;
	cout << "    a  b  c  d  e  f  g  h" << endl;
	for (int i = 0; i < 8; i++) {
		cout << "  ";
		for (int k = 0; k < 25; k++) cout << "-";
		cout << endl << i + 1 << " |";
		for (int j = 0; j < 8; j++) {
			if (desk[i][j] > 10) {
				SetColor(Blue);
			}
			else {
				SetColor(Brown);
			}
			cout << setw(2) << figure(desk[i][j] % 10);
			SetColor(Red);
			cout << "|";
		}
		cout << endl;
	}
	cout << "  ";
	for (int k = 0; k < 25; k++) cout << "-";
}

void moving() {
	char *c = new char[30];
	while (true) {
		system("cls");
		print();
		cout << "\nВведіть координати фігури для прорахування ходів(endgame - для
виходу) : ";
		gets_s(c, 35);
		if (strcmp("endgame", c) == 0) break;
		int i, k;
		if (coord_t(c, i)) {
			k = i % 10 - 1;
			i /= 10; i -= 1;
			if (desk[i][k] == 0)
				cout << "На обраному полі немає фігур";
			cout << "Можливі ходи обраної фігури (" << getname(i, k) << "):\n";
			switch (desk[i][k]) {
			case 1: pishak(i, k); break;
			case 2: kin(i, k); break;
			case 3: slon(i, k); break;
			case 4: tura(i, k); break;
			case 5: ferz(i, k); break;
```

```
                }
            }
            else
                cout << "Неправильно введені координати. Спробуйте ще раз.";
            system("pause > nul");
        }
}

void pishak(int i, int k) {
        if (desk[i + 1][k] != 0) cout << "Обрана фігура не може здійснювати хід.\n";
        if (i == 6 && desk[7][k] == 0)
            cout << "Пішак наступним ходом на " << getcoord(k, i) << '-' << getcoord(k, i +
1)
            << " може перетворитись на будь-яку фігуру\n";
        else if (i == 1 && desk[2][k] == 0 && desk[3][k] == 0)
            cout << "Можливий хід " << getcoord(k, i) << '-' << getcoord(k, i + 2) << endl;
        if (desk[i + 1][k] == 0)
            cout << "Можливий хід " << getcoord(k, i) << '-' << getcoord(k, i + 1) << endl;
}

void kin(int i, int k) {
        bool no_move = true;
        kin_t(i, k, i - 1, k - 2, no_move);
        kin_t(i, k, i + 1, k - 2, no_move);
        kin_t(i, k, i + 2, k - 1, no_move);
        kin_t(i, k, i + 2, k + 1, no_move);
        kin_t(i, k, i + 1, k + 2, no_move);
        kin_t(i, k, i - 1, k + 2, no_move);
        kin_t(i, k, i - 2, k + 1, no_move);
        kin_t(i, k, i - 2, k - 1, no_move);
        if (no_move)
            cout << "Обрана фігура не може здійснювати хід";
}

void kin_t(int i0, int k0, int i, int k, bool& n) {
        if (i >= 0 && i < 8 && k >= 0 && k < 8 && desk[i][k] == 0) {
            cout << "Можливий хід " << getcoord(k0, i0) << '-' << getcoord(k, i) << endl;
            n = false;
        }
}

void slon(int i0, int k0) {
        for (int i = i0 + 1, j = k0 - 1; i < 8 && i >= 0 && j < 8 && j >= 0; i++, j--) {
            if (desk[i][j] != 0) break;
            else cout << "Можливий хід " << getcoord(k0, i0) << '-' << getcoord(j, i) <<
endl;
        }
        for (int i = i0 + 1, j = k0 + 1; i < 8 && i >= 0 && j < 8 && j >= 0; i++, j++) {
            if (desk[i][j] != 0) break;
            else cout << "Можливий хід " << getcoord(k0, i0) << '-' << getcoord(j, i) <<
endl;
        }
        for (int i = i0 - 1, j = k0 - 1; i < 8 && i >= 0 && j < 8 && j >= 0; i--, j--) {
            if (desk[i][j] != 0) break;
            else cout << "Можливий хід " << getcoord(k0, i0) << '-' << getcoord(j, i) <<
endl;
        }
        for (int i = i0 - 1, j = k0 + 1; i < 8 && i >= 0 && j < 8 && j >= 0; i--, j++) {
            if (desk[i][j] != 0) break;
            else cout << "Можливий хід " << getcoord(k0, i0) << '-' << getcoord(j, i) <<
endl;
        }
}
```

```
void tura(int i0, int k0) {
	for (int i = i0 + 1; i < 8 && i >= 0; i++) {
		if (desk[i][k0] != 0) break;
		else cout << "Можливий хід " << getcoord(k0, i0) << '-' << getcoord(k0, i) <<
endl;
	}
	for (int j = k0 + 1; j < 8 && j >= 0; j++) {
		if (desk[i0][j] != 0) break;
		else cout << "Можливий хід " << getcoord(k0, i0) << '-' << getcoord(j, i0) <<
endl;
	}
	for (int j = k0 - 1; j < 8 && j >= 0; j--) {
		if (desk[i0][j] != 0) break;
		else cout << "Можливий хід " << getcoord(k0, i0) << '-' << getcoord(j, i0) <<
endl;
	}
	for (int i = i0 - 1; i < 8 && i >= 0; i--) {
		if (desk[i][k0] != 0) break;
		else cout << "Можливий хід " << getcoord(k0, i0) << '-' << getcoord(k0, i) <<
endl;
	}
}

void ferz(int i, int k) {
	slon(i, k);
	tura(i, k);
}

void second_task() {
	clean();
	int black = 0, white = 0, temp, var;
	random_fill(false);
	random_fill(true);
	for (int i = 0; i < 8; i++)
		for (int j = 0; j < 8; j++) {
			temp = desk[i][j];
			switch (temp%10) {
			case 0: case 6: var = 0;  break;
			case 1: var = 1; break;
			case 2: case 3: var = 3; break;
			case 4: var = 5; break;
			case 5: var = 10; break;
			}
			if (temp > 10) black += var;
			else if (temp > 0) white += var;
		}
	print();
	cout << "\n\nЧорні : " << black << endl << "Білі : " << white << endl;
	cout << "У " << ((black > white) ? "Чорних" : "Білих") << " перевага +" << abs(white -
black) << endl;
	system("pause > nul");
}

void random_fill(bool black) {
	std::srand(time(0));
	int var, rnd, rndi, rndj;
	for (int i = 1; i < 7; i++) {
		switch (i) {
		case 1: var = 8; break;
		case 5: case 6: var = 1; break;
		case 2: case 3: case 4: var = 2; break;
		}
		rnd = rand() % var;
		if (i == 6) rnd = 1;
```

```
            for (int j = 0; j < rnd; j++) {
                  do { rndi = rand() % 8; rndj = rand() % 8; } while (desk[rndi][rndj] !=
0);
                  desk[rndi][rndj] = black * 10 + i;
            }
      }
```

```
Король розміщений(на) на полі e1
Пішак розміщений(на) на полі b2
Ферзь розміщений(на) на полі b3
Тура розміщений(на) на полі f3
Слон розміщений(на) на полі c4
Кінь розміщений(на) на полі a5
Пішак розміщений(на) на полі c7
    a  b  c  d  e  f  g  h
  -------------------------
1 |  |  |  |  |Кр|  |  |  |
  -------------------------
2 |  |п |  |  |  |  |  |  |
  -------------------------
3 |  |ф |  |  |  |т |  |  |
  -------------------------
4 |  |  |С |  |  |  |  |  |
  -------------------------
5 |К |  |  |  |  |  |  |  |
  -------------------------
6 |  |  |  |  |  |  |  |  |
  -------------------------
7 |  |  |п |  |  |  |  |  |
  -------------------------
8 |  |  |  |  |  |  |  |  |
  -------------------------
Введіть координати фігури для прорахування ходів(endgame - для виходу) : b2
Можливі ходи обраної фігури (Пішак):
Обрана фігура не може здійснювати хід.
```

```
}
```

1. ПЕРЕЛІК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

Рекомендована література: основна

1. Вендеров А.М. Проектирование программного обеспечения экономических информационных систем. – М.: Финансы и статистика, 2000. – 347 с.
2. Орлов С.А. Технологии разработки программного обеспечения. – Спб.: Питер, 2002. – 463 с.
3. Соммервил И. Инженерия программного обеспечения. 6-е издание. – М.– Спб.–Киев,– 2002. –623 с.
4. Лаврищева Е.М., Коротун Т.М. Построение процесса тестирования программных систем // Проблемы программирования. – 2002. – № 1–2. – С. 272–281.
5. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб: Питер, 2001. – 368 с.
6. Лаврищева Е.М. Методы программирования. Теория, инженерия, практика. Киев: Наукова думка, 2006.–451с.
7. Corbin J. The art of Distributed Application. Programming Techn. For Remote Procedure Calls. – Berlin: Springer Verlag, 1992. – 305 .
8. Бабенко Л.П., Лаврищева Е.М. Основи програмної інженерії. – Киев.: Знання. – 2001.– 269 с.
9. Чернецки К., Айзенекер У. Порождающее программирование. Методы, инструменты, применение. – Издательский дом «Питер». – М.: СПб.– Харьков – Минск .– 2005 – 730 с.
10. Wilson C., Maples B., Londgrave T. Принципы проектирования и разработки программного обеспечения .–Учебный курс.–Пер.с анг.–М.: Русская редакция, 2000.–559с.
11. Кулаков А.Ю. Оценка качества программ ЭВМ. – Киев: Техніка. – 1984. – 167 с.
12. Андон Ф.И., Коваль Г.И., Коротун Т.М., Лаврищева Е.М., Суслов В.Ю. Основы качества программных систем. – Киев: Академпериодика.– Второе изд.– 2007. – 669 с.
13. Джалота П. Управление программными проектами на практике. – Лори, 2005.– 265 с.
14. Леффингуэлл Д., Уидриг Д. Принципы работы с требованиями к программному обеспечению. Унифицированный поход, М.: «Вильямс», 2002. – 448с.
15. Брауде Э. Технология разработки программного обеспечения. СПб.: Питер, 2004. – 687с.

Рекомендована література: додаткова

1. Вигерс К.И. Розробка вимог до ПЗ. – М.: Російська редакція Microsoft, 2004. – 575 с.

2. Шлеер С. , Меллор С. Объектно-ориентированный анализ: моделирование мира в состояниях .– Киев: Диалектика, 1993. – 240 с.
3. Кендалл Скотт. Унифицированный процесс. Основные концепции. – М.: СПБ. – Киев. – 2002. – 157 с.

Для нотаток

Для нотаток

НАВЧАЛЬНЕ ВИДАННЯ ІНСТРУКТИВНО-МЕТОДИЧНІ МАТЕРІАЛИ
ДЛЯ ЛАБОРАТОРНИХ ЗАНЯТЬ

з дисципліни «Основи програмної інженерії (CS106)»

для студентів напряму підготовки 6.050103 – Програмна інженерія.

Укладач: Бойко Ігор Володимирович